FIG. 1A (Prior Art)

← 10

```
┌─────────────────────────────────────────┐
│ Prepare HDL code (with no                │  ← 12
│ attempt at floorplanning in the code)    │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐  ← 14
│ Compile HDL code to generate RTL         │
│ net list                                 │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐  ← 16
│ perform logic optimization on RTL        │
│ net list                                 │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐  ← 18
│ map optimized RTL net list to            │
│ target architecture to generate          │
│ technology specific netlist              │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐  ← 20
│ assign specific portions of technology   │
│ specific netlist to specific portions    │
│ of an IC's area                          │
└─────────────────────────────────────────┘
                    │
                    ▼
┌─────────────────────────────────────────┐  ← 22
│ use place and route tools within each    │
│ area to create circuitry in each         │
│ area                                     │
└─────────────────────────────────────────┘
```

30 FIGS

FiG. 1B (Prior Art)   25

Prepare HDL code for first preselected
region of an IC   26

Prepare HDL code for second preselected   28
region of the IC

prepare interconnect HDL code for   30
interconnects between two regions of
the IC

perform separately synthesis for each   32
region
    (for each region, compile HDL code,
      optimize logic and map to
        target architecture)
and perform synthesis of interconnect
    HDL code

use place and route tools within each   34
region to create circuitry in each
    region

| DISPLAY 120 | | RANDOM ACCESS MEMORY 104 | READ ONLY MEMORY 106 | MASS STORAGE DEVICE 107 |

| KEYBOARD 121 |

| CURSOR CONTROL 122 |

| INPUT/OUTPUT INTERFACE 123 |

BUS 101

PROCESSOR 102

100

FIG. 2

FiG. 3   ⬐201

Prepare HDL code   ⬐203

Compile HDL Code to generate   ⬐205
technology independent RTL netlist

Perform partition of technology   ⬐207
independent RTL netlist between ICs
and/or perform floorplanning of the
RTL netlist by allocating a portion
of the RTL netlist to a portion of
an IC

optionally optimize logic in RTL   ⬐209
Netlist

map (optimized) RTL netlist to   ⬐211
target architecture to generate
technology specific netlist

use place and route tools to   ⬐213
create design of circuitry in
target architecture

FIG. 4A   ↙ 301

| Prepare HDL code | 303 |

↓

| Compile HDL code to generate technology independent RTL netlist | 305 |

↓

| Optionally perform Hierarchical Interconnect Optimization | 307 |

↓

| Optionally perform Hierarchical Resource Estimation | 309 |

↓

| perform partitioning of technology independent RTL netlist between ICs (optionally perform replication of logic from one IC to another IC and/or optionally split a unitary RTL component into first and second portions of the RTL component on two ICs) | 311 |

↓

| optionally optimize logic in RTL netlist | 313 |

↓

| map RTL netlist to target architecture to create design of circuitry in target architecture | 315 |

↓

| use place and route tools to implement circuitry in target architecture | 317 |

FIG. 4B

351

Prepare HDL code — 353

Compile HDL code to generate technology independent RTL netlist — 355

Optionally perform Hierarchical Interconnect Optimization — 357

Optionally perform Hierarchical Resource Estimation — 359

Perform floor planning using the technology independent RTL netlist by allocating a portion of the RTL netlist to a portion of an IC (optionally also perform partitioning and/or replicating and/or splitting) — 361

optionally optimize logic in RTL netlist — 363

map RTL netlist to target architecture to generate technology specific netlist — 365

use place and route tools to create design of circuitry in target architecture — 367

FIG. 5A

Hierarchical Interconnect Optimization ~401

Examine Interconnects between RTL
modules resulting from HDL
compilation ~403

perform Interconnect optimization
(e.g. remove duplicative I/O's between
modules) at technology
independent RTL netlist level ~405

FIG. 5.B

411

416

HDL module 2 (RTL level)

424

419

HDL module 1 (RTL level)

30

421

"1"

418

422

logic

420

415

425

50

HDL module 3 (RTL level)

417

FIG. 6

501

User selects target technology
(target architecture used by
specific vendor (e.g. Xilinx) to
implement logic functions in on
IC such as an FPGA)

503

System performs mapping of RTL
netlist for each module (separately)
to target architecture

505

System specifies logic and I/O
resource estimates to implement each
module in target architecture and
specifies timing estimates for
each module after mapping

507

User may consider these resource
and timing estimates in deciding
how + whether to partition
and/or allocate/floorplan and/or
replicate and/or split

509

FIG. 7A

← 601

User (or system automatically) directs a partitioning (and/or floorplanning and/or replicating and/or splitting) by selecting/assigning an RTL netlist module to another IC (e.g. a 2nd IC) ⟍ 603

System replicates signals between/among ICs (e.g. common input signals) ⟍ 605

System creates new RTL netlist for each IC (e.g. new I/O pins, logic etc.) ⟍ 607

FIG. 7B

615

620

621

IC 1

IC 2

619

617

618

RTL netlist module 1

RTL netlist module 2

622

623

624

Fig. 8A

701

```verilog
module prep2_2 (DATA0, DATA1, DATA2, LDPRE, SEL, RST, CLK, LDCOMP);
output [7:0] DATA0 ;
input [7:0] DATA1, DATA2;
input LDPRE, SEL, RST, CLK, LDCOMP;
    wire [7:0] DATA0_internal;
    prep2_1 inst1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0_internal);
    prep2_1 inst2 (CLK, RST, SEL, LDCOMP, LDPRE, DATA0_internal, DATA2, DATA0);

endmodule
```

— 703

```verilog
module prep2_1 (CLK, RST, SEL, LDCOMP, LDPRE, DATA1, DATA2, DATA0);
input CLK, RST, SEL, LDCOMP, LDPRE ;
input [7:0] DATA1, DATA2 ;
output [7:0] DATA0;
reg [7:0] DATA0;
reg [7:0] highreg_output, lowreg_output; // internal registers

wire compare_output =    (DATA0 == lowreg_output);   // comparator
wire [7:0]  mux_output =    SEL ? DATA1 : highreg_output;   // mux

// registers
always @ (posedge CLK  or posedge RST)
begin
    if (RST) begin
        highreg_output = 0;
        lowreg_output = 0;
    end else begin
            if (LDPRE)
            highreg_output = DATA2;
            if (LDCOMP)
            lowreg_output  = DATA2;
    end
end

// counter
always @(posedge CLK or posedge RST)
begin
    if (RST)
        DATA0 = 0;
    else if (compare_output)   // load
        DATA0 = mux_output;
    else
        DATA0 = DATA0 + 1;
end

endmodule
```
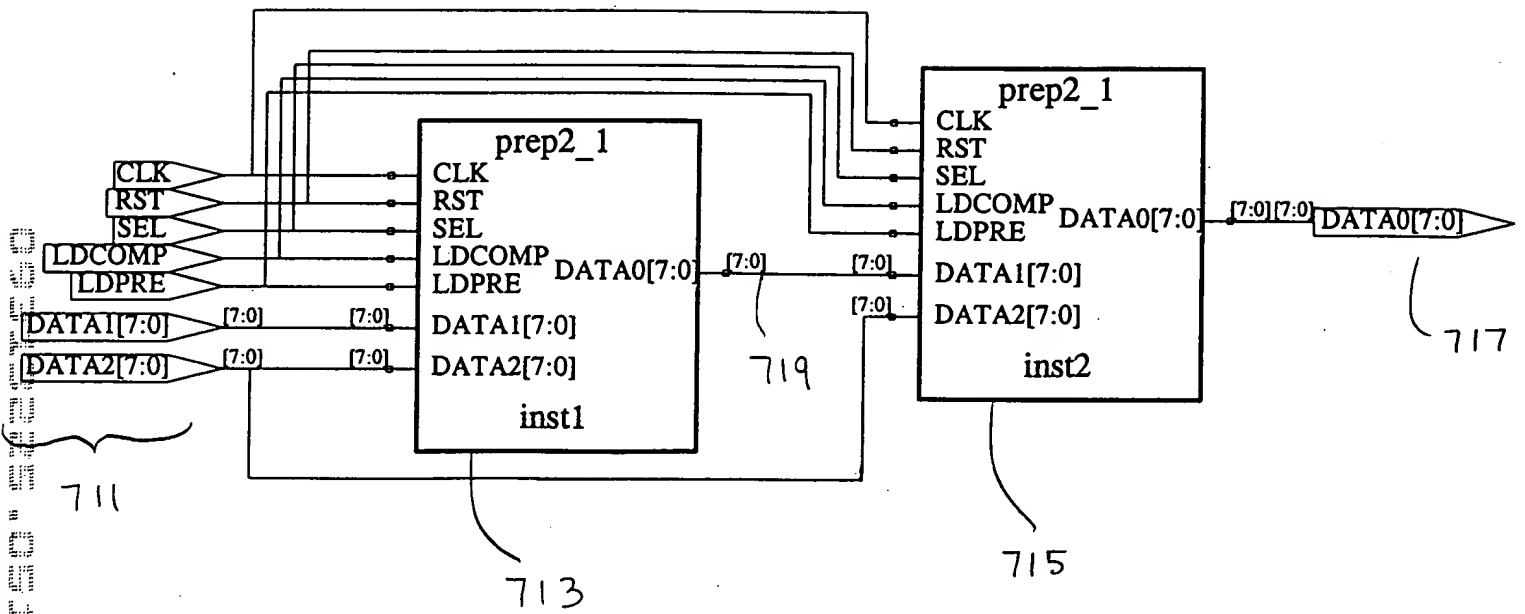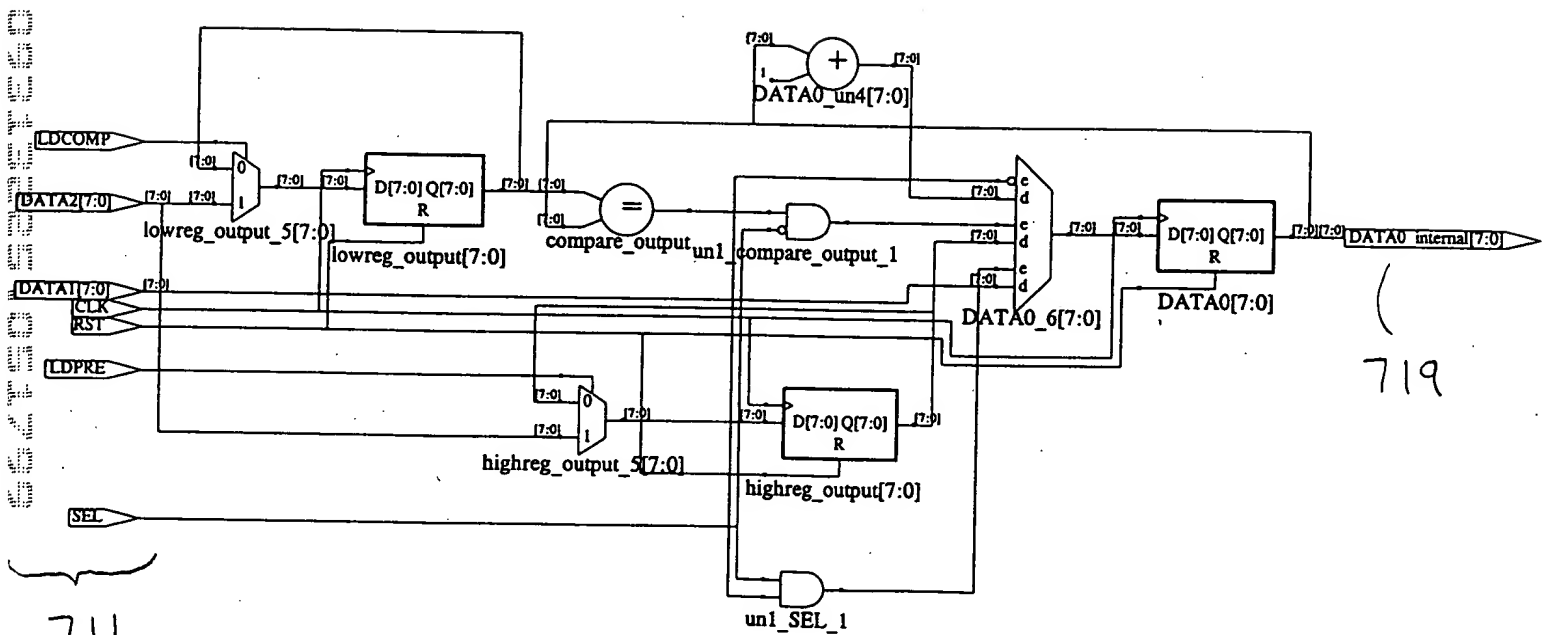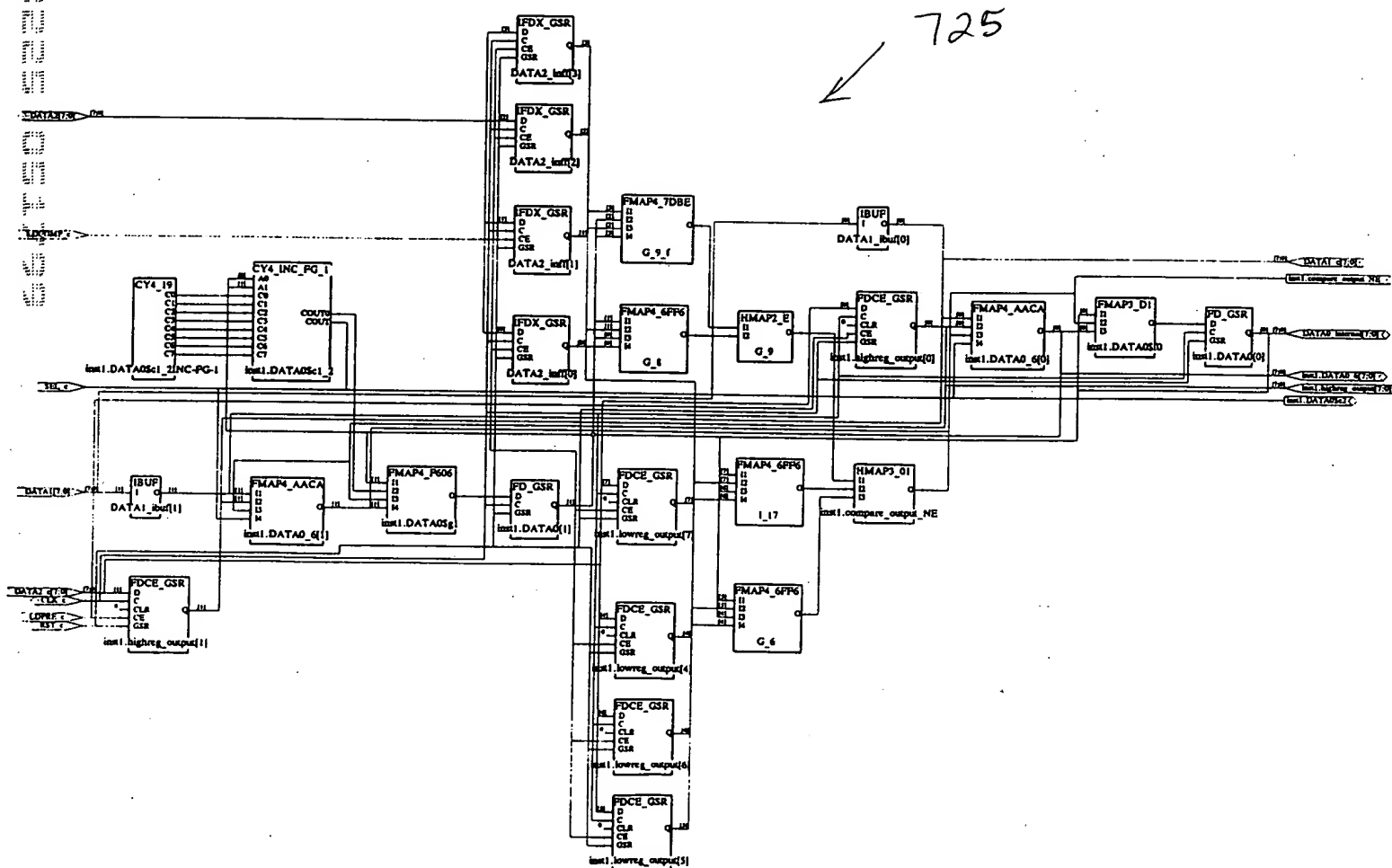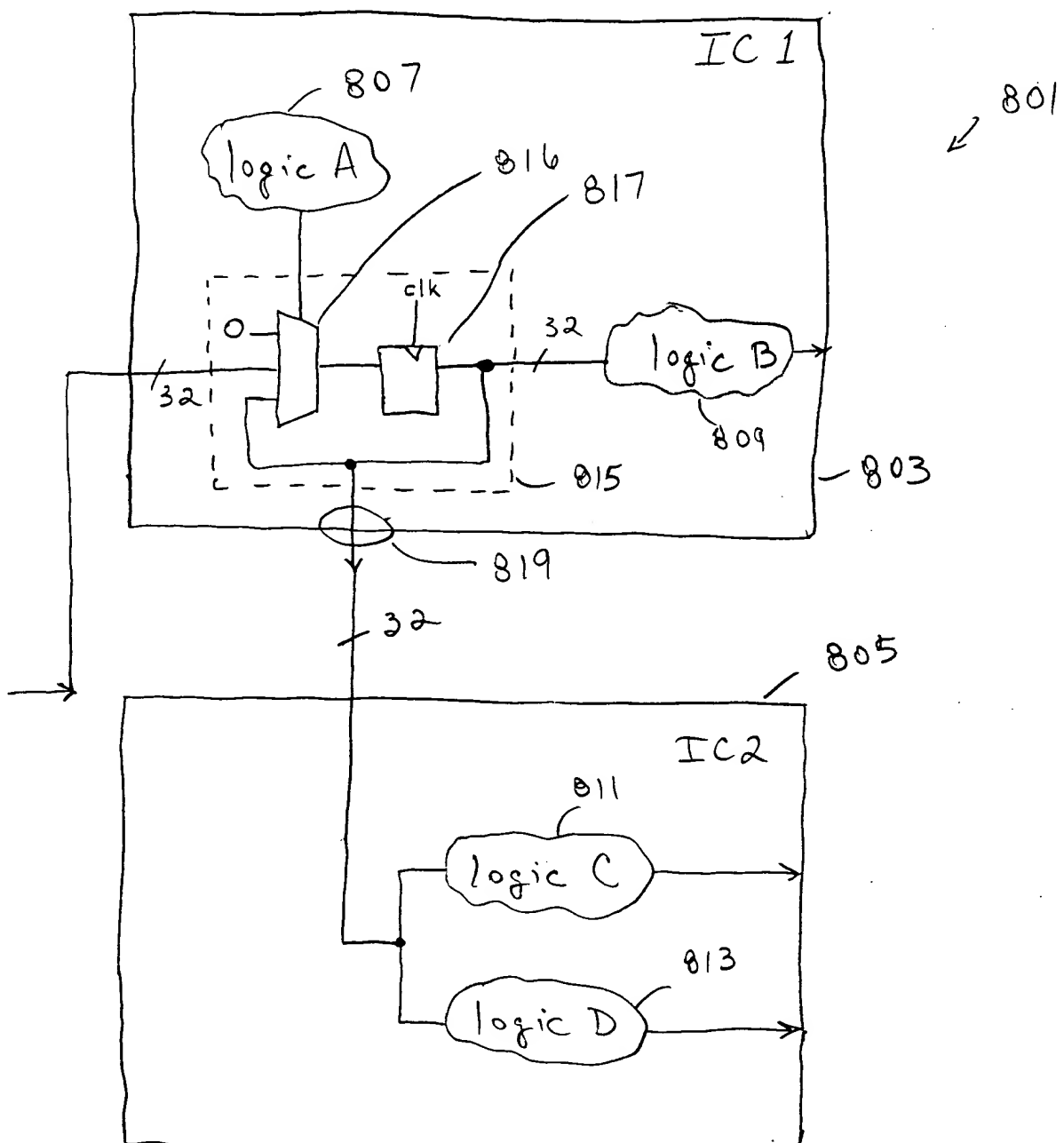
— 705

FIG. 8B

FIG. 8 C

FIG. 8D

FIG. 9 A

IC 1

807

logic A

816

817

clk

O

32

logic B

809

32

815

803

819

32

805

IC2

811

logic C

813

logic D

801

FIG. 9B

FIG. 9C

FPGA 1 — 835

4 → 4 to 16 Decoder → 16

833

831

FPGA 2 — 837

FPGA 3 — 839

FIG. 9D

4 to 16 Decoder   FPGA 1 — 835A

X 4

4 →   4 to 16 Decoder   FPGA 2 — 837A

4 →   4 to 16. Decoder   FPGA 3 — 839A

FiG. 9E

Floor planned region A.

logic A

floor planned region B

logic B

851
852
854
853
855

858
857
856
859

Fig. 9F

Floorplanned region A

D Q

logic A

floorplanned region B

D Q

logic B

852
854
856
853A
855

857
859A
856A

FIG. 10 A

901

IC

logic A — 905

logic B — 907

12 → Rom

Addr   Data → 100

903

FIG. 10 B

905   909

logic A   IC 1

12 → Rom

Addr   Data → 50

903 A

907   911

logic B

Rom

12 → Addr   Data → 50

12

903 B

FIG. 10 C

921

logic A

923

927 A 32
929 B 32

Adder
"+"

925

R [31:0]
32

933

Cin

931

FIG. 10 D

923   935

931   logic A

Cin

927A A[15:0] 16
929A B[15:0] 16

Adder
"+"

925A

16 R [15:0]

933A

931A   939

927B

A[31:16] 16
929B B[31:16] 16

Adder
"+"

925B

16 R [31:16]

933B

FIG. 10E

Select large RTL object(s) in technology independent RTL Netlist (after compiling HDL code to generate this Netlist ─ 951

Split large RTL object into several RTL objects and determine several RTL objects can remain on same IC ─ 953

can remain

cannot remain on same IC

split large RTL object into several RTL objects on different ICs ─ 955

957

Do not split large RTL object

FIG. 11A

1001

RTL Module A
1009

RTL Module B
1008

RTL Module C
1007

RTL Module D
1006

Available IC area

1003

1005

FIG. 11B

1001

RTL Module A
1009

RTL Module B
1008

RTL Module C
1007

RTL Module D
1006

RTL modules B+C

1003C

RTL Module A

RTL Module D

1003B

1003A

1005

1003

FIG. 12

IC 1 1021

1027

logic A 1027

clock 1029

1031

1025

IC 2 1023

logic B 1028

1032

clock 1030

FIG. 13 1051

1052

'HDL Source for a design' 1055

Technology Independent RTL Netlist 1057

1053

HDL Compiler 1061

Partitioning Routinues 1067

Technology mapping Routinues 1063

Floorplanning routinues 1069

Replication routinues 1071

Optimization Routinues 1065

splitting routinues 1073